Personalización de Aplicaciones en OO-H

Cristina Cachero[1], Irene Garrigós[1], and Jaime Gómez[1]

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante. SPAIN
{ccachero,igarrigos,jgomez}@dlsi.ua.es

Abstract La inclusión de políticas de personalización en el marco de métodos hipermediales influye en todos los niveles de modelado de una aplicación web. Desde OO-H mantenemos que esta política de personalización debe facilitar la definición de aspectos no conocidos *a priori*, lo que a su vez obliga a estos métodos a posibilitar la adaptación dinámica de las aplicaciones ante cambios imprevistos en el entorno. Este artículo presenta, en el contexto del Método Hipermedial Orientado al Objeto (*Object Oriented Hypermedia (OO-H)*), un marco de personalización que cumple estas características y que se vertebra en torno a cuatro elementos fundamentales: (1) un conjunto de actividades de diseño que captura la parte estática de la personalización, (2) un mecanismo de especificación de reglas, definidas en base a una plantilla XML, que constituye el modelo de personalización dinámico (3) una arquitectura de ejecución que permite el desacoplamiento de este modelo dinámico respecto al resto de modelos y minimiza el efecto de cambios en la estrategia, y (4) un repositorio extensible que incluye un conjunto de mecanismos de registro de la actividad del usuario en el sistema. La posibilidad de extensión de este repositorio facilita su adaptación a las características particulares de la aplicación modelada.

1 Introducción

Es un hecho reconocido que la necesidad de manejar los riesgos inherentes a la creciente complejidad de las aplicaciones demandadas por las organizaciones para su explotación en Internet ha sido el principal detonante de la aparición, en los últimos años, de un amplio rango de métodos, técnicas y procesos ingenieriles que, englobados bajo el epígrafe común de Ingeniería Web [12], se orientan a facilitar la comprensión, desarrollo, evolución y mantenimiento de las aplicaciones web. Este esfuerzo ha supuesto en muchos casos la extensión de técnicas aplicadas en la ingeniería del software 'tradicional' con nuevos constructores y vistas hipermediales [15] que abordan el problema de la navegación del usuario a través del espacio de información.

Sin embargo, la idiosincrasia de la web introduce una serie de nuevos retos que van mucho más allá del planteamiento de este mapa de navegación, y que incluyen aspectos como necesidad de evolución continua, margen de error cero, disponibilidad global de fuentes de información o tratamiento de una audiencia heterogénea y, a menudo, imposible de definir *a priori* [10, 18, 24]. La comunidad hipermedial ha aceptado dichos retos y en la actualidad asistimos a una evolución de las principales propuestas de modelado (ver e.g. [8, 15, 25]) para dar respuesta a cada uno de los requisitos de usuario existentes. En concreto, la inclusión de estrategias (estáticas o dinámicas) de identificación de perfiles que modelan los distintos tipos de audiencia y la consecuente personalización de los entornos ofertados se ha convertido en uno de los aspectos que más interés ha despertado desde el punto de vista de los métodos de modelado hipermedial, debido al efecto que las políticas de personalización tienen sobre todo el ciclo de desarrollo de la aplicación, desde la recogida de requisitos a la propia arquitectura de ejecución.

En este contexto, este artículo presenta la respuesta que el Método Hipermedial Orientado al Objeto (*Object Oriented Hypermedia (OO-H)* [13]) proporciona a dicho problema. Desde OO-H, y siguiendo la filosofía presentada en [14], abogamos por una ejecución de las actividades de diseño de la personalización de manera paralela al resto de actividades de modelado. Para ello el artículo está estructurado como sigue: en la sección 2 se presenta una visión general de las técnicas de personalización existentes en la actualidad, en términos de fuentes de información utilizadas, principales

tipos de algoritmos y su efecto sobre la capacidad de adaptación de las aplicaciones resultantes. En la sección 3 se detalla cómo los métodos de modelado hipermedial en general y OO-H en particular han incorporado esas técnicas en el marco de un proceso genérico de desarrollo de aplicaciones, y se presenta el efecto que esta incorporación causa en las distintas dimensiones de modelado de una aplicación web. A continuación, en la sección 4 se introducen, con la ayuda de un ejemplo, las actividades de personalización que OO-H incorpora en tiempo de diseño. Estas actividades se basan en un marco de información extensible que almacena información referente tanto a la actividad del usuario en el sistema como a las condiciones de la interacción, y que, junto a su estrategia de poblamiento, se presenta en la sección 5. Este marco también sirve de soporte para el almacenamiento de políticas de personalización dinámicas, i.e. definibles en tiempo de ejecución mediante la especificación de un conjunto de reglas de activación que se expresan en base a una plantilla XML, cuya estructura y semántica será mostrada en la sección 6.

2 Contexto de la Personalización

La personalización de aplicaciones, entendida como la capacidad de alteración dinámica con el fin de proporcionar al usuario la impresión de estar trabajando con una aplicación específicamente diseñada para dar satisfacción a sus necesidades particulares (también denominado por algunos autores *One to One Delivery* [8]), se puede producir en base a tres tipos principales de fuentes de información [8, 10, 14]:

- **Datos preexistentes**, normalmente importados de fuentes externas
- El **perfil y las preferencias** de cada usuario individual, que pueden ser proporcionadas de forma explícita por el propio usuario o recogidas por la aplicación de manera implícita, a través de la actividad del usuario en el sistema
- El **contexto** en el que se produce la interacción usuario-aplicación (localización, momento, características de la red etc), que se recoge (si está disponible) de forma implícita por la aplicación como parte integrante de esa interacción

La integración o no de cada una de estas fuentes de datos en el seno de una aplicación concreta determina el conjunto de técnicas que pueden ser aplicadas con el fin de implementar la política de personalización deseada. En la actualidad, el número y variantes de estas técnicas parece incontable [1]. Sin embargo, y con el fin de facilitar la discusión, las hemos clasificado en cinco grandes grupos [21, 22]:

- **Filtrado Simple**. El Filtrado Simple consiste en la definición de grupos de usuarios, a los que se asocian vistas particulares de la aplicación. Estos grupos de usuarios pueden estar predefinidos (e.g. en base a roles), o bien crearse en base a determinados atributos del usuario o del contexto (URL de conexión, género, edad...). Como ejemplo, se puede diseñar una vista restringida de la aplicación destinada a niños, y determinar que cualquier usuario menor de 12 años acceda a esa vista. Otro ejemplo clásico es el análisis de la URL para determinar el idioma en que se debe presentar la aplicación.
- **Reglas de Asociación**. Esta técnica supone la definición, normalmente en forma de ficheros externos que son consultados por la aplicación en tiempo de ejecución, de criterios que, cuando son cumplidos por un determinado usuario y (opcionalmente) ante la ocurrencia de un determinado evento, desencadenan una acción. En este artículo nos referiremos a **Reglas Reactivas** cuando el evento sea directamente causado por el propio usuario (e.g. activación de un enlace en la interfaz), y hablaremos de **Reglas Proactivas** cuando dicho evento tenga que ver con un cambio en el entorno de interacción y/o en el estado de la aplicación. Un ejemplo clásico se da en los sistemas de comercio electrónico, donde a menudo se definen reglas de productos complementarios que son ofrecidos ante la compra del producto principal. La principal ventaja de esta técnica es la disminución del acoplamiento (con la consiguiente disminución del coste de mantenimiento y evolución) de las actividades de personalización respecto al resto de actividades del método. Como inconveniente podemos citar la necesidad de establecer mecanismos de resolución de conflictos en caso de contradicción. La estructura y comportamiento concreto de las reglas en el marco de OO-H será visto en la sección 6.

- **Filtrado por contenido**. En este tipo de técnica, y partiendo de un conjunto de categorías, se determina la pertenencia o no de los objetos a cada una de dichas categorías, y se muestran aquellos que entren en el grupo de categorías de interés de cada usuario. Esta técnica tiene como principal inconveniente el gran grado de objetividad que requiere. No obstante, se suele utilizar en determinados tipos de sistemas, como son los sistemas de clasificación de documentos.
- **Filtrado colaborativo**. Esta técnica, que permite aplicar criterios subjetivos y por tanto es muy utilizada como soporte de sistemas de recomendación (ver e.g. Amazon [2]), se basa en la actividad de otros usuarios de comportamiento similar al usuario actual para determinar aspectos como cuál será la próxima acción que el usuario realice en el sistema o en qué productos puede estar interesado. El filtrado colaborativo está en el origen del concepto de la Navegación Social [26], que pretende simular los mecanismos de orientación de los que dispone el usuario en el mundo real. Esta técnica, que proporciona una capacidad de adaptación elevada a la aplicación, tiene como principales inconvenientes el alto coste computacional al que puede dar lugar en presencia de un alto volumen de usuarios y el bajo grado de fiabilidad de los algoritmos ante un volumen de información reducido, lo que obliga a establecer estrategias alternativas mientras no se disponga de dichos datos.
- **Perfilado** (*Profiling*). Por último, existen técnicas que, a partir de un identificador proporcionado por el propio usuario de manera implícita, permiten la personalización de la aplicación en base a la actividad del usuario en otras aplicaciones web. El uso de este identificador global limita los problemas asociados a la seguridad y privacidad de datos personales. Sus principales inconvenientes son, por un lado, la necesidad de que el propio usuario obtenga y proporcione este identificador global, y por otro la necesidad de intercambio e integración de esa información proviniente de fuentes externas para su posterior procesado.

Además de las ventajas e inconvenientes mencionados en cada técnica, otros aspectos que hay que tener en cuenta a la hora de decidirse por una u otra forma de personalización incluyen consideraciones relativas a (1) los costes (en términos tanto monetarios como de eficiencia y escalabilidad), (2) estrategias de privacidad y seguridad de los datos y (3) control de la propia adaptación, en términos de quién la autoriza, en qué momento se realiza y cómo se puede deshacer en caso de no adecuarse a las necesidades del usuario.

En primer lugar, hay que evaluar el coste monetario asociado a determinadas estrategias, que requieren el uso de complejos algoritmos de minería de datos (*Data Mining*). Por otro lado es necesario evaluar el tiempo de respuesta de las aplicaciones personalizadas ante incrementos en el número de usuarios. Este parámetro es fundamental para determinar aspectos como la granularidad de la información analizada, el método y la frecuencia de análisis de datos o incluso cuestiones relativas a la arquitectura, como es el uso de servidores de caché. Además, debido al tipo de información manejada, es necesario asegurar al usuario que el conocimiento (explícito o implícito) que tiene el sistema sobre él no va a ser accedido, difundido ni utilizado para otros fines que el facilitar y agilizar la tarea del propio usuario en el sistema. En este sentido existen estándares de seguridad e intercambio de datos como son el CPEX (Customer Profile EXchange [9]) o el P3P (Platform for Privacy Preferences Project [29]) que pueden ser integrados para la consecución de este objetivo. Por último, se tiene que especificar el modo en que se va a controlar la personalización de la interfaz. Cuestiones como qué hacer cuando se infiere una nueva regla de adaptación (si debe ser validada por un administrador o incluida como parte de la política de manera automática), en qué orden se deben ejecutar dichas reglas y qué hacer si existe una contradicción, cuándo materializar el cambio (inmediatamente o la próxima vez que el usuario entre al sistema) o qué mecanismos ofertar para que el usuario pueda corregir o evitar una personalización no deseada (e.g. mediante la eliminación del conocimiento previo, con carácter temporal o permanente, para adaptarse a las necesidades del usuario en la presente sesión [28]) son vitales a la hora de establecer una política global de personalización.

Para finalizar esta sección nos gustaría destacar cómo el efecto que la elección de una u otra técnica tiene sobre la aplicación se materializa en la manera en que se recoge la información necesaria, y define su capacidad de adaptación, dando lugar a su clasificación en tres grandes grupos:

adaptables, adaptivas o proactivas. Las aplicaciones adaptables (*Customizable* [14]), realizan el proceso de personalización en base a información introducida por el propio usuario en el sistema. Suelen utilizar por tanto técnicas de filtrado simple, filtrado por contenidos o profiling. Las aplicaciones adaptivas ((*Adaptive*)[14] utilizan por el contrario información recogida y analizada por la propia aplicación de manera transparente para el usuario. Ejemplos típicos de este tipo de aplicaciones son los sistemas de recomendación utilizados como apoyo en numerosas aplicaciones de comercio electrónico, que utilizan mecanismos de filtrado colaborativo y reglas de asociación. Por último las aplicaciones proactivas (*Proactive*) [10] son capaces de detectar *de motu propio* cambios en el entorno y reaccionar en consecuencia, actuando de este modo como Observadoras [11] del entorno. La principal diferencia respecto de las aplicaciones adaptivas es que, mientras que en éstas cualquier cambio es causado por una acción del usuario, una aplicación proactiva puede cambiar la vista del usuario sin necesidad de que éste realice ninguna acción. Implica por tanto tecnología *push*, frente al tradicional esquema *pull* de las aplicaciones web. Un ejemplo de comportamiento proactivo es la capacidad por parte de la aplicación de detectar un cambio en las condiciones de interacción (por ejemplo en la localización de un dispositivo de acceso móvil), y la modificación automática de la vista de información ofertada en función de dicha localización.

Una vez identificado el problema, a continuación presentamos las respuestas que, tanto desde el ámbito de la comunidad hipermedial como de otras áreas, se han propuesto, y que están en la base de nuestra aproximación.

3 Personalización e Hipermedia

El reto de la personalización, lejos de ser una innovación de la comunidad hipermedial, ha sido un tema de discusión desde que se empezó a considerar al usuario como eje alrededor del cual debía girar el desarrollo de aplicaciones interactivas. Numerosos grupos de investigación provinientes de áreas tan dispares como la Inteligencia Artificial o la Interacción Usuario-Máquina han presentado incontables propuestas (ver e.g. [3, 5, 6, 19]) que, haciendo uso de tecnologías tan dispares como las bases de datos, cookies, generación dinámica de páginas, algoritmos de aprendizaje computacional, técnicas de descubrimiento de patrones, inferencia basada en reglas o minería de datos, son capaces de desarrollar interfaces inteligentes, preparadas para predecir el comportamiento del usuario en el sistema y facilitar la consecución de sus objetivos individuales [1, 14].

Sin embargo ya hemos comentado cómo la aparición del hipertexto ha dado un nuevo sentido a esta personalización. En este tipo de aplicaciones, la necesidad de fidelizar al usuario ha provocado un esfuerzo adicional para cubrir las necesidades particulares de estructuración navegacional, así como para dar soporte a contextos de uso altamente volátiles en términos de disparidad de dispositivos, imprevisibilidad de la red o ubicuidad y atemporalidad del usuario. A nivel empresarial este esfuerzo se ha materializado en la aparición de nuevos consorcios (ver e.g. el Consorcio de Personalización [22]) y estándares para la descripción tanto para definir preferencias de usuario y capacidades de dispositivo (ver CC/PP [29]) como para asegurar el intercambio, la privacidad y seguridad de la información que posibilita la personalización (ver e.g. CPEX o P3P). Además, se ha desarrollado un gran número de herramientas comerciales (e.g. ILog JRules, WebSphere, Rainbow, WebPlaces o LikeMinds, por citar algunas) que facilitan el uso de las técnicas y estrategias de personalización y que por tanto actualmente dan soporte a numerosas aplicaciones web personalizables.

En este contexto, el bajo nivel de abstracción al que se están definiendo las políticas de personalización (en las que se incluye la integración de estos estándares y aplicaciones) está causando problemas de falta de reuso y dificultad de mantenimiento y escalabilidad de las aplicaciones personalizadas resultantes, lo cual ha provocado que algunas propuestas de modelado hipermedial (ver e.g. [7, 16, 23]) estén evolucionando para incluir técnicas y marcos específicos de modelado de personalización que faciliten su posterior implantación y mantenimiento.

3.1 El papel de la personalización en OO-H

Existe un acuerdo general en cuanto al papel que la personalización juega dentro de las dimensiones de modelado de aplicaciones hipermediales [14], que, con las extensiones de OO-H, puede ser vista en la Fig. 1.

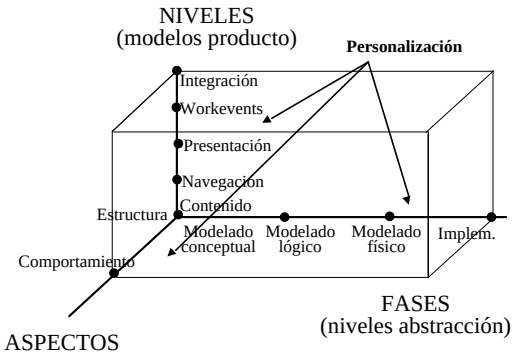

Figure1. *Dimensiones de Modelado de una Aplicación Hipermedial*

En esta figura se observa cómo los distintos rasgos que intervienen en el modelado de una aplicación hipermedial se pueden desdoblar en tres dimensiones. La primera dimensión comprende cinco *Niveles* en los que OO-H desglosa las características que intervienen en el modelado de una aplicación web. A los tres tradicionales (contenido, navegación y presentación), OO-H añade otros dos. Por un lado, el análisis del flujo de trabajo de la aplicación permite determinar las variaciones en su estado en función del momento de ejecución en que se encuentre dicha aplicación, y en consecuencia cómo puede variar la percepción que el usuario tiene de la misma (e.g. qué caminos de navegación o qué opciones de funcionalidad están disponibles en función de ese momento de ejecución). Por otro, el nivel de integración permite definir arquitecturas que incluyen módulos de funcionalidad preexistentes, y especificar cómo el resto de la aplicación debe interactuar con ellos. La dimensión de *Aspectos* por su parte refleja las vistas tradicionales (estructura y comportamiento) que intervienen en cualquiera de los niveles. La tercera dimensión, por último, presenta las distintas *Fases* de un ciclo de desarrollo de software, desde análisis hasta implementación, y es ortogonal a las otras dos.

La inclusión de consideraciones relativas a la personalización como parte de las actividades de diseño afecta a todo el marco de trabajo. Como ejemplo, a nivel de modelado conceptual de comportamiento de contenido (fase, aspecto y nivel respectivamente, ver Fig. 1) o, lo que es lo mismo, durante la actividad de modelado conceptual de la lógica de dominio, las actividades de personalización pueden inducir la inclusión del patrón Strategies [11] como parte de dicho modelo. Este patrón permite definir una familia de algoritmos, encapsular cada uno de ellos, y convertirlos en intercambiables en función del usuario y/o de su contexto de interacción. Si consideramos el modelado lógico de la estructura de la navegación del usuario a través del espacio de información, las consideraciones de personalización pueden causar una modificación en estos caminos de navegación. De este modo, la personalización dota a la aplicación de una capacidad de alteración dinámica que le permite proporcionar al usuario la impresión de estar trabajando con una aplicación específicamente diseñada para dar satisfacción a sus necesidades particulares, lo que algunos autores denominan difusión uno a uno (*one-to-one delivery* [8]).

Desde OO-H pensamos que la existencia en este tipo de propuestas de un modelo conceptual proporciona una oportunidad única de plantear estrategias de personalización basadas en la semántica

asociada a los átomos de información y no en su estructuración física. Como ejemplo, un modelo conceptual nos permite distinguir las páginas de acceso de las páginas de información. Ante una estrategia de generación de atajos a las páginas más visitadas, el esquema conceptual posibilita que esas páginas de acceso no sean candidatas a aparecer en la lista final. Además, el modelado y diseño de la personalización dentro de un marco conceptual permite entender mejor los mecanismos utilizados, y descubrir rasgos comunes que posibilitan el reuso de patrones, componentes, algoritmos o incluso subsistemas [23].

Esto no impide que, para extraer el máximo provecho a estas características, sea necesaria la integración en estos modelos hipermediales de técnicas y algoritmos avanzados provinientes de otras áreas de conocimiento. OO-H, concebido desde sus orígenes como un método orientado al usuario [17] y dotado de capacidades de interconexión con módulos preexistentes, proporciona mediante su nivel de integración un marco ideal para la interacción no sólo con módulos de lógica de negocio sino también con algoritmos y técnicas de personalización (ver Fig. 1).

Antes de pasar a detallar los distintos mecanismos que ofrece OO-H, y con el fin de ilustrar los conceptos que iremos presentando a lo largo del artículo, supongamos que tenemos un sistema de información simplificado, expresado en UML [27] y que representa la reserva de habitaciones de una cadena de hoteles (ver Fig. 2).

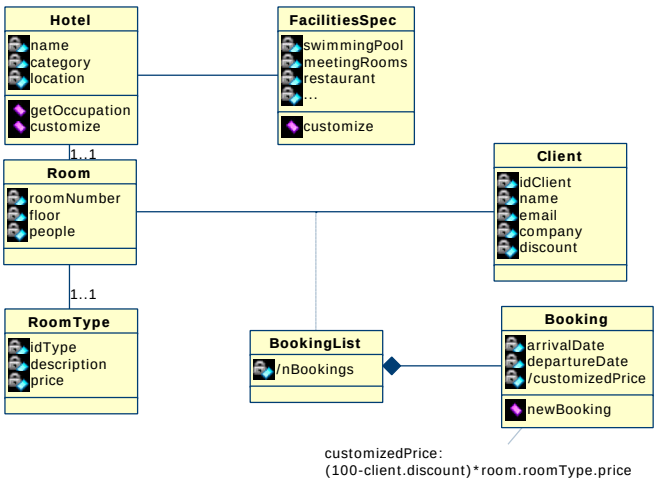

Figure2. *Diagrama de Clases del Hotel*

Esta cadena dispone de diversos hoteles, cada uno con un nombre, una localización y una categoría, así como una serie de servicios asociados (clase *FacilitySpec*): piscina, restaurante, salas de reuniones... Cada hotel dispone además de una serie de habitaciones que pueden ser reservadas por clientes registrados. La barra inclinada frente al precio de alquiler de la habitación (atributo *customizedPrice* de la clase *Booking*) es la notación que utiliza UML para indicar que el atributo en cuestión es un atributo derivado (calculado en este caso a partir del precio oficial y el descuento aplicable al cliente que realiza la reserva). Por claridad, vamos a suponer que nos interesa modelar la vista del rol 'recepcionista' dentro de esta aplicación. Además, por razones de espacio, en este artículo nos centraremos en las estructuras necesarias para proporcionar un marco conceptual de

personalización de contenido y estructuras de navegación, dejando de lado otros aspectos, también importantes, como son la personalización de presentación, flujo de trabajo, integración de componentes o incluso personalización de la propia personalización.

Ya hemos comentado cómo el proceso de modelado de la interfaz de este sistema se ve afectado desde sus fases más tempranas por las capacidades de personalización que deseemos incluir. En este sentido, OO-H incluye dos posibilidades: el modelado de personalización en tiempo de diseño, que será presentado a continuación, y el modelado en tiempo de ejecución, que será detallado en la sección 6.

4 Modelado de Personalización en tiempo de diseño en OO-H

Las estrategias de personalización más sencillas y estables son aquellas basadas en información y reglas de negocio conocidas en tiempo de diseño. Estas técnicas, que proporcionan una personalización 'enlatada', predefinida en el cuerpo de la propia interfaz, están presentes en la mayoría de métodos de modelado hipermedial [8, 15, 25] y son suficientes para la mayoría de las aplicaciones web existentes en la actualidad. En OO-H su definición afecta a los siguientes rasgos:

- Su propio proceso de diseño.
- Definición de atributos derivados y filtros durante la fase de diseño navegacional.
- Aplicación de patrones de Personalización en cualquier fase del ciclo de desarrollo de la aplicación.

En relación con el Proceso de Diseño, OO-H se define como una aproximación dirigida por los requisitos de usuario. La categorización de estos usuarios en perfiles estáticos da lugar a distintos diagramas de navegación (DAN) en función de estos perfiles.

Volviendo a nuestro ejemplo, supongamos que el recepcionista necesita gestionar las reservas del hotel. En la Fig. 3 observamos dos Destino Navegacionales (DN), cuya notación es el símbolo de paquete en UML. OO-H define el DN como un constructor que permite la estructuración del modelo en niveles, con lo que se gestiona la complejidad de los diagramas. Cada DN agrupa los caminos navegacionales y las vistas de información que dan respuesta a un subconjunto de requisitos de usuario. Por el momento, nos centraremos en el DN *Booking Management*, y aplazaremos la explicación del DN *Customization* hasta el final de la presente sección, cuando abordemos el uso de patrones en OO-H. En la Fig. 4, correspondiente a la explosión del DN *Booking Management*, observamos cómo dicho DN contiene el modelado de la manera en que el diseñador ha decidido que la aplicación dé respuesta al requisito de gestión de reservas anteriormente mencionado.

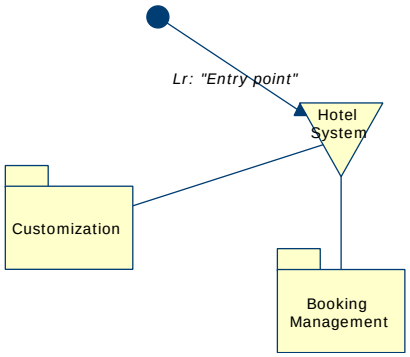

Figure3. *Diagrama Navegacional Nivel 0*

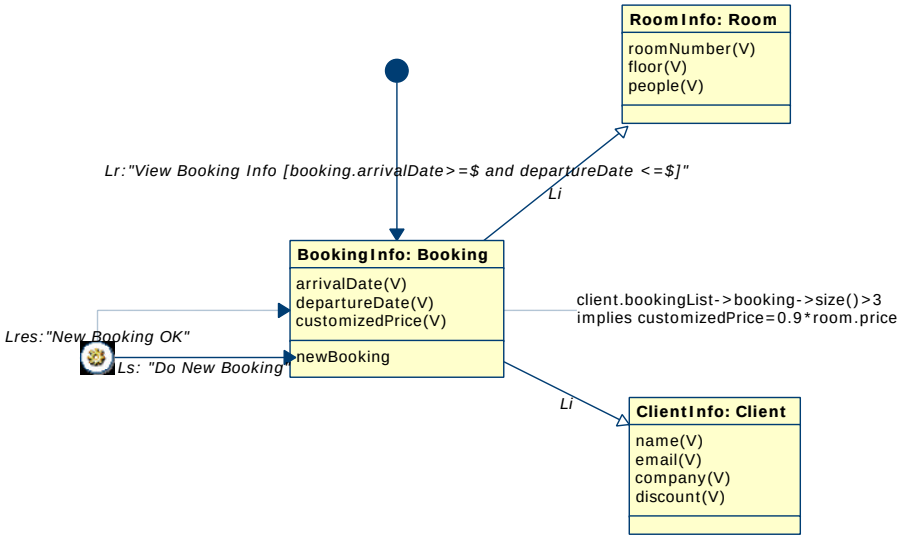

Figure4. *Destino Navegacional Booking Management*

Más concretamente, en la Fig. 4 observamos cómo el recepcionista tiene acceso a la lista de reservas comprendidas entre dos fechas introducidas en tiempo de ejecución. Este hecho se refleja en la restricción (o *filtro*, tal y como se denomina en OO-H) *arrivalDate >= $ and departureDate <= $*, asociada al enlace *View Booking Info* y donde el símbolo *$* sustituye a un valor que debe ser introducido por el usuario en tiempo de ejecución. En el mismo diagrama también observamos cómo esta lista de reservas incluye información tanto acerca de la reserva en sí (fecha de llegada, fecha de partida y precio) como acerca de la habitación y el cliente involucrados en dicha reserva. Desde esa vista, el recepcionista también puede realizar una nueva reserva, hecho que se modela mediante el enlace de servicio *Do New Booking*.

Aunque este proceso de diseño permite un primer nivel de personalización en base a tipos predefinidos de usuario, a menudo es necesario incluir condiciones de personalización más detalladas, que en OO-H son modeladas mediante el uso de atributos derivados y filtros. Para la definición de estos atributos derivados y filtros OO-H incluye el Lenguaje de Restricciones de Objeto (*Object Constraint Language (OCL)* [27]) como lenguaje de consulta, lo cual permite el uso de características almacenadas bien sea en el propio modelo conceptual del sistema, bien sea en el repositorio que forma parte del marco de personalización que proporciona OO-H, y que será detallado en la sección 5. De este modo se puede especificar tanto el valor de determinados átomos de información como las características de los caminos de navegación o incluso la forma en la que se debe invocar a los métodos en función de las características individuales del usuario.

Volviendo a nuestro ejemplo, observamos cómo la definición de un precio personalizado en función del número de reservas que el cliente ha realizado en el sistema (ver atributo *customizedPrice* y regla OCL asociada en la Fig. 4) es un ejemplo trivial de personalización de contenido.

Por último, OO-H incluye un mecanismo de patrones para facilitar el refinamiento de los modelos [13]. Dentro de estos patrones, y de cara a la personalización, tienen especial relevancia el Patrón de Identificación (que genera una página de petición de usuario y contraseña) y el Patrón de Preferencias. El patrón de preferencias recoge la práctica común en muchas aplicaciones de pedir al usuario que restrinja su conjunto de objetos de interés mediante la cumplimentación de un formulario. Es por ello que se aplica sobre elementos del modelo de navegación (atributos y/o clases navegacionales). Su uso afecta al modelo navegacional, tal y como podemos ver en las Figs. 3 y 5. En el primer caso, la aplicación del patrón crea un nuevo requisito de usuario (introducir valores de personalización),

que se materializa en el nuevo DN *Customization* (ver Fig. 3). Dentro de este DN (ver Fig. 5), el constructor de tipo Colección (cuya representación gráfica es un triángulo invertido) *My Hotel* agrupa los átomos de información afectados por el patrón (en nuestro ejemplo el atributo *hotel.category* y todos los atributos de la clase *FacilitiesSpec*), y permite que el usuario introduzca los valores que desea mediante la activación de los métodos *customize* asociados. Estos métodos también son introducidos de manera automática como consecuencia de la aplicación del patrón.

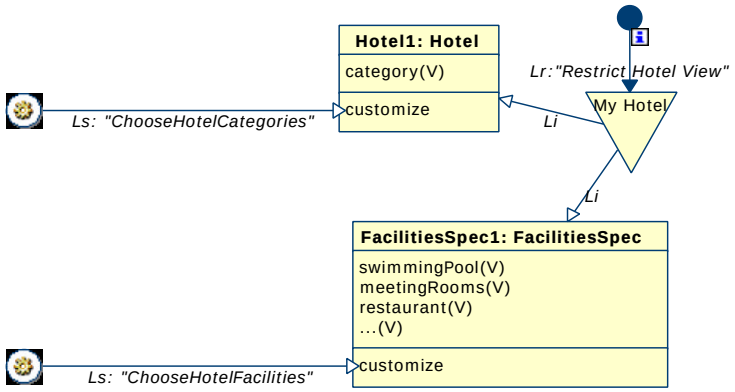

Figure5. Explosión del DN *Customization*

Como el lector ya habrá deducido de lo expuesto hasta el momento, para la implementación de estas políticas de personalización es necesario proporcionar un marco de información sobre el que definir las distintas estrategias. A continuación presentamos dicho marco, que da soporte no sólo a las técnicas de modelado en tiempo de diseño sino también a las técnicas en tiempo de ejecución que serán presentadas en la sección 6.

5 El repositorio de Personalización en OO-H

Sea cual sea el tipo de estrategia de modelado de la personalización (en tiempo de diseño o en tiempo de ejecución), ésta debe ser definida en función de una serie de estructuras de información que proporcionen, o al menos permitan calcular, los distintos perfiles de usuario definidos como parte de esa estrategia. Esta información se almacena en OO-H en un repositorio (ver Fig. 6) que estructura el conjunto inicial de átomos de información sobre los que se puede establecer la política de personalización deseada. El diseñador puede incluir y conectar este marco de trabajo con cualquier modelo de interfaz OO-H al que quiera dotar de capacidad de personalización. A partir de ahí, OO-H permite la extensión de dicho repositorio con las características particulares (como puede ser la inclusión de nuevos algoritmos) que requiera la aplicación.

El repositorio, a pesar de no ser un marco cerrado, sí refleja sin embargo los tres grandes conceptos que intervienen en el modelado de la personalización en OO-H, y que son (1) perfiles de usuario, (2) información de contexto y (3) reglas de asociación.

Por un lado tanto los datos proporcionados por el propio usuario como la información recogida a partir de su actividad en el sistema determinan el Perfil de Usuario. Este perfil puede capturar aspectos muy diversos: preferencias, patrones de comportamiento (búsquedas realizadas, páginas visitadas, productos o servicios adquiridos...), características, problemas técnicos, etc. Es importante notar que, aunque esta información puede ser proporcionada explícitamente por el propio usuario mediante la definición de un simple formulario de entrada (como ocurría cuando aplicábamos el Patrón de Personalización en el ámbito de OO-H), es un hecho reconocido que este mecanismo es un método invasivo debido a que requiere un cambio en el foco de atención del usuario, lo que a su vez causa que en la mayoría de los casos éste no personalice [28]. Es por ello que actualmente

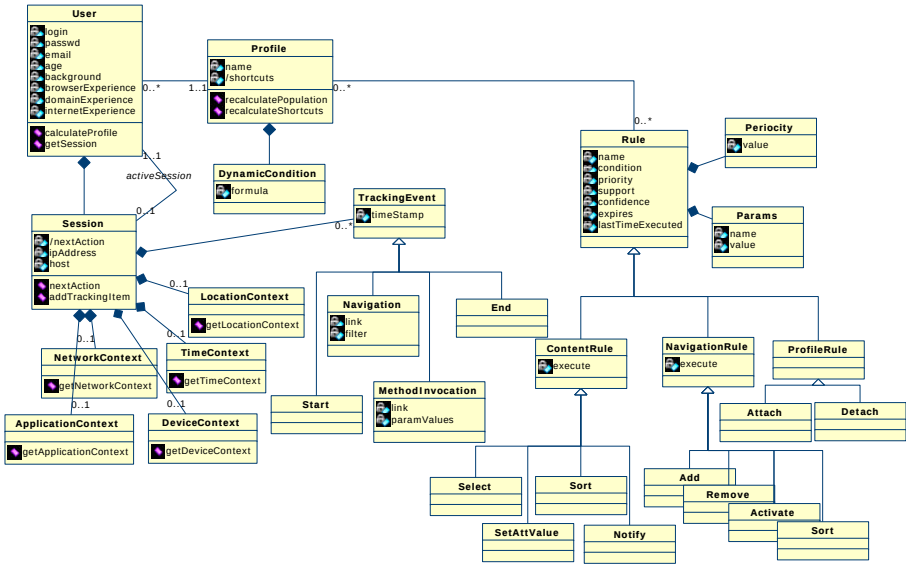

Figure6. *Marco de trabajo de Personalización*

la tendencia generalizada es intentar deducir estos perfiles a partir de información que puede ser obtenida de manera transparente al usuario, tal y como comentábamos en la sección 1.

Volviendo a la Fig 6, en ella podemos observar cómo el perfil de usuario se almacena en la clase *User*, que contiene un conjunto de objetos *Session*, uno por cada sesión que haya abierto el usuario en el sistema. El usuario puede estar preasignado a un perfil (clase *Profile*) o ser asignado dinámicamente en cualquier momento durante su sesión. En cualquier caso, y con el fin de proporcionar información sobre la actividad del usuario en el sistema a los algoritmos de filtrado, OO-H define cuatro tipos de evento de usuario, correspondientes a las acciones de usuario que pueden ser registradas actualmente de manera automática en OO-H. Este registro se produce mediante la instanciación de distintos tipos de objetos de seguimiento (*TrackingItems*, ver Fig. 6), a saber:

- Start: implica la entrada de un nuevo usuario en el sistema.
- Navigation: implica la activación de un enlace de navegación en la interfaz asociada. Una acción de navegación guarda su contexto, es decir, las condiciones de filtro que tenía asociadas cuando se activó.
- Method Invocation: implica la invocación de un servicio ofertado por la lógica. Este tipo de acción supone guardar tanto el enlace de servicio activado como los valores introducidos en los parámetros para esa invocación.
- End: implica la salida del usuario del sistema.

La inserción de estos objetos como parte de la sesión del usuario se realiza mediante la asociación de eventos de creación de objetos de seguimiento a los distintos tipos de enlace que ofrece OO-H. En nuestro ejemplo, el enlace de entrada a la aplicación, etiquetado como *Entry Point* en la Fig. 4, lleva asociado un evento de creación de un objeto de seguimiento de tipo *start*, que se lanza cada vez que un nuevo usuario activa el enlace de entrada a la aplicación. Del mismo modo, la activación del enlace de servicio *Do New Booking* crea un objeto de seguimiento de tipo *MethodInvocation*, que almacena en su atributo *link* el id del enlace activado. Siguiendo con la misma filosofía, la acción de pinchar (activar) cualquier enlace de navegación en OO-H provoca, paralelamente al efecto de navegación, la creación de un objeto de seguimiento de tipo *Navigation*, que almacena, además del enlace en cuestión, la información de contexto (objeto concreto desde el cual se navegó, restricciones en la población destino del enlace etc) de dicha navegación. Por último, la instanciación de un objeto de seguimiento de tipo *End* (que, como efecto colateral, finaliza la sesión del usuario en el sistema) se

puede producir de dos modos: bien por la activación de un enlace definido como 'de salida' en OO-H (*Exit Point*) o bien de manera automática, cuando la aplicación detecta un período de inactividad en el sistema superior a un valor umbral (en este momento establecido en 5 minutos). Los ítems de seguimiento son especialmente útiles para la detección de nuevos patrones de navegación de usuario, como puede verse en [20]. Además, y debido a que en OO-H los ítems de seguimiento no guardan URL's sino enlaces conceptuales (ver Fig. 6), las facilidades que ofrecen los métodos de modelado hipermedial de una categorización exhaustiva de estos enlaces (por ejemplo identificación de determinados enlaces de servicio como enlaces de tipo 'compra', o de determinados enlaces de navegación como enlaces de tipo 'descarga' [4]) permite definir estrategias mucho más precisas.

Todas las acciones dentro de una sesión están implícitamente ordenadas por la marca de tiempo que llevan asociadas. Esta información y su orden son la base para diversos algoritmos, que quedan fuera del ámbito de discusión de este artículo. La inclusión de estos algoritmos en el marco de trabajo de OO-H se produce mediante la asociación de servicios que capturan su interfaz a las distintas clases de personalización. En la Fig. 6 podemos ver cómo el modelado del sistema de reserva de hoteles sobre este marco incluye tres algoritmos de personalización como parte del modelado de la estrategia: (1) predicción de la siguiente acción del usuario (*aSession.recalculateNextAction()*), (2) reasignación de un usuario a un perfil de manera dinámica en función de los valores de los atributos y/o caminos de actividad registrados en el marco de trabajo (*aUser.calculateProfile*) y (3) generación de un menú personalizado que recoge los destinos más frecuentados del grupo al que pertenece al usuario (*aProfile.recalculateShortcuts*).

El repositorio ofrece además estructuras para el almacenamiento del contexto en el que se produce la interacción usuario-aplicación. Este contexto se incluye como parte de la sesión del usuario, y en OO-H se vertebra en torno a cinco ejes (ver Fig. 6):

- Contexto de red: latencia, ancho de banda etc de la conexión.
- Contexto Temporal: fecha y hora local de la conexión
- Contexto de Dispositivo: tipo de dispositivo (palm, portátil, PC...) y software instalado en él.
- Contexto de Localización: ubicación del cliente
- Contexto de Aplicación: estado de la aplicación

Obviamente, cada aplicación usará el subconjunto de información que considere relevante (o al que tenga acceso). Este tipo de información es muy utilizado para el modelado de reglas proactivas, tal y como veremos en la sección 6.

Por último, el repositorio también proporciona una estructura de almacenamiento de las reglas de personalización. Estas reglas, que pueden provocar un cambio en cualquier nivel de la aplicación (ver Fig. 1), son presentadas a continuación.

6 Modelado de personalización en tiempo de ejecución en OO-H

El uso de técnicas de modelado de presentación en tiempo de diseño como las presentadas en la sección 4 tienen como principal desventaja el alto acoplamiento con el resto de la aplicación. Este tipo de aproximación resulta por tanto poco flexible ante la necesidad de variaciones frecuentes en la estrategia de personalización para adaptarse a las condiciones variables del entorno, tal y como ocurre en aplicaciones de comercio electrónico. Es por ello que OO-H, siguiendo la estela de otras aproximaciones [8], aboga por incorporar un mecanismo de externalización del modelado de las estrategias de personalización en forma de reglas de activación.

OO-H define estas reglas mediante una plantilla XML [30]. La arquitectura de ejecución de las aplicaciones generadas a partir de modelos OO-H permite que esta plantilla pueda ser modificada y reprocesada por la aplicación sin necesidad de recompilar el resto de módulos. Tal y como vemos en la Fig. 6, estas reglas se categorizan, en función del resultado que su activación tiene en el sistema,

en reglas de contenido (*ContentRule*), de navegación (*NavigationRule*) y de definición de perfil (*ProfileRule*).

Como ejemplo, supongamos que deseamos expresar una nueva política del hotel, que determina que, si la ocupación del hotel el día que se desea realizar una reserva es menor del 80%, el cliente se beneficia de un 10% de descuento adicional sobre el precio de la habitación. Esta regla en OO-H se expresaría como sigue:

```
<TPersonalization>
    ...
    <profile name="ooh:all">
        <rule type="content" name="DiscountOnBookings" support="5"
        confidence="100" priority="10" execute="post">
            <params>
                <param name="bookingDay" value="session.doNewBooking.params->date"/>
                <param name="cPrice" value="session.client.bookingList->
                booking.customizedPrice"
            </params>
            <event type="MethodInvocation" link="DoNewBooking" />
            <condition value="session.hotel.getOccupation(bookingDay)<80"/>
            <action type="setAttValue" value="cPrice=cPrice*0.9" notify="yes"/>
        </rule>
        ...
    </profile>
    ...
</TPersonalization>
```

En ella podemos ver cómo los distintos elementos XML corresponden a clases/atributos del marco presentado en la Fig. 6. En primer lugar, observamos cómo una regla se define en el contexto de un perfil. En nuestro ejemplo, y para indicar que la regla se debe evaluar sin importar el usuario que entre en el sistema, a dicha etiqueta se le ha asociado el valor "ooh:all", predefinido en el contexto de OO-H. Las reglas de acción tienen además varios atributos asociados, y que son [21]:

- Nombre: nombre de la regla
- Tipo de regla: tipo de efecto que la activación de la regla causaría en el sistema. En nuestro ejemplo se trata de una regla que modifica el valor de un atributo, por lo que es de tipo *Content*.
- Soporte: número de veces que se ha activado la regla en nuestra aplicación.
- Confianza: número de veces que ha acertado en su ejecución. El tipo de política que representa esta regla hace presuponer que el usuario siempre va a estar de acuerdo con ella.
- Prioridad: importancia de la regla en el sistema en caso de conflicto o condiciones de carrera.
- Ejecución: su valor determina si la ejecución de la regla se debe realizar con anterioridad o posterioridad a la ejecución de la acción de usuario.

Cada regla puede tener asociada una sección de parámetros, que en OO-H actúan como variables locales y simplifican el resto de expresiones.

Por su parte el cuerpo de la regla lo constituyen tres secciones. En primer lugar el Evento *Event* determina qué acción de usuario actúa como disparador de la regla. De manera alternativa, se puede establecer una Periocidad de la regla (*Periocity*, ver Fig. 6), que especifica cada cuánto tiempo, de manera automática, la aplicación debe chequear dicha regla. En el contexto de la etiqueta de periocidad, OO-H define un valor especial, *ooh:always*, que indica que el cumplimiento de las condiciones de activación se debe comprobar de manera continua. Es precisamente este concepto de Periocidad lo que permite implementar el concepto de Interfaces Proactivas del que hablábamos en la sección 1.

La segunda parte del cuerpo de la regla lo constituye la condición, que especifica una expresión de guardia que permite o inhibe la ejecución de la regla. En nuestro ejemplo el valor asociado a la etiqueta *condition* establece que la ocupación del día para el cual el cliente desea realizar una reserva

debe ser menor de un 80%.

Por último la etiqueta de acción (ver Fig. 6) puede tomar el valor de cualquiera de los tipos correspondientes al tipo de regla de que se trate. En nuestro caso, al tratarse de una regla de contenido, es correcto definir una acción de tipo *SetAttValue*.

7 Conclusiones y trabajos futuros

En el presente artículo hemos presentado el conjunto de mecanismos proporcionados por OO-H que posibilitan la definición de una política de personalización como parte del proceso de modelado de este tipo de aplicaciones. Esta integración se produce en base a un marco de personalización que incluye tres elementos fundamentales. En primer lugar se define una serie de actividades en tiempo de diseño que permiten el modelado de la parte invariable de la política de personalización, i.e. aquella que no se prevé que cambie a pesar de la ocurrencia de posibles variaciones en el entorno. Estas actividades se complementan con una arquitectura capaz de importar en tiempo de ejecución (i.e. sin necesidad de recompilar ningún módulo de la aplicación) un conjunto de reglas de activación, expresadas mediante plantillas XML, que definen la parte dinámica (variable) de dicha política. Además, OO-H incluye un repositorio que estructura la información necesaria para implementar la política de personalización deseada. La integración de este repositorio en el modelo conceptual de la aplicación, unido a la capacidad de interconexión con módulos preexistentes en OO-H posibilita la extensión de dicho repositorio con nuevos átomos de información y/o algoritmos en función de las necesidades particulares de la aplicación. El marco de personalización en OO-H proporciona además los mecanismos necesarios para la captura de la actividad del usuario en el sistema.

En este momento nos encontramos en fase de integración de los mecanismos dinámicos (reglas de acción) en la herramienta CASE que da soporte a OO-H. En esta primera fase, la implementación se está restringiendo a Reglas Reactivas, i.e. activadas a partir de alguna acción del usuario en el sistema. Esperamos que esta implementación nos permita evaluar la eficiencia de la aproximación, y valorar su potencia semántica en base a su aplicación a sistemas reales. Otra área de trabajo importante es el soporte de estándares como CPEX o P3P (aún no incluidos en OO-H), y que nos permitirían capturar información proviniente del usuario o incluso de otras aplicaciones web en este formato.

References

[1] *Personalization*, volume 43, 08 2000.

[2] Web de Amazon. www.amazon.co.uk, 2002.

[3] C. Avery and R. Zeckhauser. Recommender Systems for Evaluating Computer Messages. In *Communications of the ACM*, 03 1997.

[4] M. Bieber, H. Oinas-Kukkonen, and V. Balasubramanian. Hypertext Functionality. *ACM Computing Surveys*, 31(4), 12 1999.

[5] P. De Bra. Design Issues in Adaptive Web-Site Development. In *2nd Workshop on Adaptive Systems and User Modeling on the WWW*, 1999.

[6] J. Carroll and A. Aaronson. Learning by Doing With Simulated Intelligent Help. In *Communications of the ACM*, 09 1988.

[7] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modeling language for designing Web sites WWW9 Conference. In *First ICSE Workshop on Web Engineering, International Conference on Software Engineering*, 05 2000.

[8] S. Ceri, P. Fraternali, and S. Paraboschi. Data-Driven One-to-One Web Site Generation for Data-Intensive Applications. In *Prooceedings of the VLDB'99*, 02 1999.

[9] Customer Profile EXchange. www.cpexchange.org, 2001.

[10] P. Fraternali. Tools and Approaches for Developing Data-Intensive Web Applications: a Survey. *ACM Computing Surveys*, 2000.

[11] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
[12] A. Ginige and S. Murugesan. Web Engineering: an Introduction. *IEEE Multimedia Special Issue on Web Engineering*, pages 14–18, 04 2001.
[13] J. Gómez, C. Cachero, and O. Pastor. Conceptual Modelling of Device-Independent Web Applications. *IEEE Multimedia Special Issue on Web Engineering*, pages 26–39, 04 2001.
[14] G. Kappel, W. Retschitzegger, and W. Schwinger. Modeling Customizable Web Applications - A Requirement's Perspective. In *2000 Kioto International Conference on Digital Libraries*, 11 2000.
[15] N. Koch, A. Kraus, and R. Hennicker. The Authoring Process of the UML-based Web Engineering Approach. In *Proceedings of the 1st International Workshop on Web-Oriented Software Technology*, 05 2001.
[16] N. Koch and M. Wirsing. Software Engineering for Adaptive Hypermedia Applications. In *3rd Workshop on Adaptive Hypertext and Hypermedia (8th International Conference on User Modeling)*, 07 2001.
[17] J. Kramer, S. Noronha, and J. Vergo. A User-Centered Design Approach to Personalization. In *ACM Computing Surveys*, 08 2000.
[18] D. Lowe and B. Henderson-Sellers. Web Development: Addressing Process Differences. *Cutter IT Journal (Submitted)*, 2001.
[19] M. McIlhagga, A. Light, and I. Wakeman. Towards a Design Methodology for Adaptive Applications. In *ACM/IEEE International Conference on Mobile Computing and Networking*, 10 1998.
[20] B. Mobasher, R. Cooley, and J. Srivastava. Creating Adaptive Web Sites Through Usage-Based Clustering of URL's. In *IEEE Workshop on Knowledge and Data Engineering Exchange*, 11 1999.
[21] J. Pavón. Personalización de servicios en la Web. ZOCO 2001. http://tdg.lsi.us.es/ zoco/res/ppt/pavon.zip, 2001.
[22] Personalization Consortium. www.personalization.org, 2002.
[23] G. Rossi, D. Schwabe, and R. Guimaraes. Designing Personalized Web Applications. In *Tenth International World Wide Web Conference*, pages 275–284, 05 2001.
[24] R.S. Pressman. *Software Engineering. A practitioner's approach (Fifth edition)*. Mc Graw Hill, 2000.
[25] D. Schwabe, G. Rossi, and D. J. Barbosa. Systematic Hypermedia Application Design with OOHDM. In *Proceedings of the the seventh ACM conference on HYPERTEXT '96*, page 166, 1996.
[26] M. Svensson, K. Höök, J. Laaksolahti, and A. Waern. Social Navigation of Food Recipes. In *SIGCHI'01*, 03 2001.
[27] OMG Unified Modelling Language Specification. http://www.rational.com/uml/, 06 1999.
[28] D. VanderMeer, K. Dutta, and A. Datta. Enabling Scalable Online Personalization on the Web. In *ACM Conference on Electronic Commerce (EC'00)*. ACM, 10 2000.
[29] WWW Consortium. http://www.w3.org/, 2000.
[30] eXtensible Markup Language. http://www.w3.org/XML/, 2000.